

FILE ID**CHKDMO

F 7

CCCCCCCC HH HH KK KK DDDDDDDD MM MM 000000
CCCCCCCC HH HH KK KK DDDDDDDD DD MMMM MMMM 00 00
CC HH HH KK KK DD DD 00 00
CC HH HH KK KK DD DD 00 00
CC HH HH KK KK DD DD 00 00
CC HH HH KK KK DD DD 00 00
CC HHHHHHHHHH KKKKKK DD DD 00 00
CC HHHHHHHHHH KKKKKK DD DD 00 00
CC HH HH KK KK DD DD 00 00
CC HH HH KK KK DD DD 00 00
CC HH HH KK KK DD DD 00 00
CC HH HH KK KK DD DD 00 00
CC HH HH KK KK DD DD 00 00
CC HH HH KK KK DDDDDDDD MM MM 000000
CCCCCCCC HH HH KK KK DDDDDDDD MM MM 000000 . . .
CCCCCCCC HH HH KK KK DDDDDDDD MM MM 000000 . . .

LL IIIII SSSSSSS
LL IIIII SSSSSSS
LL II SS
LL LLLLLLLL IIIII SSSSSSS
LL LLLLLLLL IIIII SSSSSSS

CH
VO

```
1 0001 0 MODULE CHKDMO (
2 0002 0           LANGUAGE (BLISS32),
3 0003 0           IDENT = 'V04-000'
4 0004 0           ) =
5 0005 1 BEGIN
6 0006 1
7 0007 1
8 0008 1 ****
9 0009 1 *
10 0010 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
11 0011 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
12 0012 1 * ALL RIGHTS RESERVED.
13 0013 1 *
14 0014 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
15 0015 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
16 0016 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
17 0017 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
18 0018 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
19 0019 1 * TRANSFERRED.
20 0020 1 *
21 0021 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
22 0022 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
23 0023 1 * CORPORATION.
24 0024 1 *
25 0025 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
26 0026 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
27 0027 1 *
28 0028 1 *
29 0029 1 ****
30 0030 1
31 0031 1 ++
32 0032 1
33 0033 1 FACILITY: F11ACP Structure Level 1
34 0034 1
35 0035 1 ABSTRACT:
36 0036 1
37 0037 1     This routine dismounts the volume in use if it should be.
38 0038 1
39 0039 1 ENVIRONMENT:
40 0040 1
41 0041 1     STARLET operating system, including privileged system services
42 0042 1     and internal exec routines.
43 0043 1
44 0044 1 --
45 0045 1
46 0046 1
47 0047 1 AUTHOR: Andrew C. Goldstein, CREATION DATE: 29-Apr-1977 17:19
48 0048 1
49 0049 1 MODIFIED BY:
50 0050 1
51 0051 1     V03-026 HH0049    Hai Huang      16-Aug-1984
52 0052 1     Call IOCSDALLOC_DMT to handle deallocation on dismount.
53 0053 1
54 0054 1     V03-025 HH0047    Hai Huang      13-Aug-1984
55 0055 1     Correct IOCSDALLOC_DEV linkage (UCB address in R5).
56 0056 1
57 0057 1     V03-024 ACG0441   Andrew C. Goldstein, 9-Aug-1984 16:31
```

: 58 0058 1 Rework dismount interlocking to eliminate races and
59 0059 1 uninterlocked processing.
60 0060 1
61 0061 1 V03-023 ACG0438 Andrew C. Goldstein, 2-Aug-1984 11:39
62 0062 1 Release cache locks when deallocating volume caches;
63 0063 1 use central dequeue routine.
64 0064 1
65 0065 1 V03-022 LMP0275 L. Mark Pilant, 23-Jul-1984 14:08
66 0066 1 Don't try to delete an uninitialized ACL.
67 0067 1
68 0068 1 V03-021 CDS0004 Christian D. Saether 20-Jun-1984
69 0069 1 Temporarily raise the process diocnt around the
70 0070 1 SQIO so that it will never be blocked. Also raise
71 0071 1 ASTCNT so it will not fail for that reason.
72 0072 1
73 0073 1 V03-020 CDS0003 Christian D. Saether 8-May-1984
74 0074 1 Have UPDATE_DIRSEQ routine queue for exclusive
75 0075 1 and cancel conversion of the volume lock to invalidate
76 0076 1 the ucb dirseq counter. Do not call the routine
77 0077 1 from the check_dismount routine anymore.
78 0078 1
79 0079 1 V03-019 CDS0002 Christian D. Saether 22-Apr-1984
80 0080 1 Use routine LOCK_COUNT.
81 0081 1
82 0082 1 V03-018 ACG0415 Andrew C. Goldstein, 9-Apr-1984 10:56
83 0083 1 Interface change to ACL_DELETEACL
84 0084 1
85 0085 1 V03-017 HH0008 Hai Huang 9-Apr-1984
86 0086 1 Change R2 thru R5 to NOPRESERVE in the linkage of the
87 0087 1 EXESDEAPGDSIZ routine.
88 0088 1
89 0089 1 V03-016 LMP0221 L. Mark Pilant, 27-Mar-1984 13:39
90 0090 1 Change UCB\$L_OWNNUIC to ORB\$L_OWNER and UCB\$W_VPROT to
91 0091 1 ORB\$W_PROT.
92 0092 1
93 0093 1 V03-015 ACG0408 Andrew C. Goldstein, 23-Mar-1984 14:42
94 0094 1 Add AST parameter so that impure storage is fully based
95 0095 1
96 0096 1 V03-014 CDS0011 Christian D. Saether 3-Mar-1984
97 0097 1 Remove UNLOCK_XQP call. It is done prior to this
98 0098 1 point now. Also KILL_CACHE happens in CLEANUP.
99 0099 1
100 0100 1 V03-013 CDS0010 Christian D. Saether 10-Feb-1984
101 0101 1 Changes to deallocate AQB and buffer when last
102 0102 1 dismount occurs on it.
103 0103 1 Replace FLUSH_FID call with KILL_CACHE call.
104 0104 1
105 0105 1 V03-012 CDS0009 Christian D. Saether 29-Dec-1983
106 0106 1 Use L_NORM linkage and BIND_COMMON macro.
107 0107 1
108 0108 1 V03-011 CDS0008 Christian D. Saether 16-Oct-1983
109 0109 1 Dequeue blocking lock.
110 0110 1
111 0111 1 V03-010 CDS0007 Christian D. Saether 21-Sep-1983
112 0112 1 Release locks in final stages of dismount so that
113 0113 1 getlki check on volume lock is not confused by counting
114 0114 1 an allocation lock for this request.

: 115 0115 1
: 116 0116 1
: 117 0117 1
: 118 0118 1
: 119 0119 1
: 120 0120 1
: 121 0121 1
: 122 0122 1
: 123 0123 1
: 124 0124 1
: 125 0125 1
: 126 0126 1
: 127 0127 1
: 128 0128 1
: 129 0129 1
: 130 0130 1
: 131 0131 1
: 132 0132 1
: 133 0133 1
: 134 0134 1
: 135 0135 1
: 136 0136 1
: 137 0137 1
: 138 0138 1
: 139 0139 1
: 140 0140 1
: 141 0141 1
: 142 0142 1
: 143 0143 1
: 144 0144 1
: 145 0145 1
: 146 0146 1
: 147 0147 1
: 148 0148 1
: 149 0149 1
: 150 0150 1
: 151 0151 1
: 152 0152 1
: 153 0153 1 **
: 154 0154 1
: 155 0155 1
: 156 0156 1 LIBRARY 'SYSSLIBRARY:LIB.L32';
: 157 0157 1 REQUIRE 'SRCS:FCPDEF.B32';
: 158 1148 1
: 159 1149 1
: 160 1150 1
: 161 1151 1 Part of this routine runs at IPL\$_SYNCH, so it must be locked into the
: working set.
: 162 1152 1
: 163 1153 1
: 164 1154 1
: 165 1155 1 LOCK_CODE;
: 166 1156 1
: 167 1157 1
: 168 1158 1 FORWARD ROUTINE
: 169 1159 1 CHECK_DISMOUNT : L_NORM NOVALUE; ! check volume for dismount
: 170 1160 1 UPDATE_DIRSEQ : L_NORM; ! bump volume directory sequence count

```
: 172 1161 1 GLOBAL ROUTINE CHECK_DISMOUNT : L_NORM NOVALUE =
: 173 1162 1
: 174 1163 1 ++
: 175 1164 1
: 176 1165 1 FUNCTIONAL DESCRIPTION:
: 177 1166 1
: 178 1167 1 This routine checks if the volume in use is marked for dismount and
: 179 1168 1 idle. If so, it completes the dismount.
: 180 1169 1
: 181 1170 1 CALLING SEQUENCE:
: 182 1171 1 CHECK_DISMOUNT ()
: 183 1172 1
: 184 1173 1 INPUT PARAMETERS:
: 185 1174 1 NONE
: 186 1175 1
: 187 1176 1 IMPLICIT INPUTS:
: 188 1177 1 CURRENT_UCB: UCB of unit in use
: 189 1178 1 CURRENT_VCB: VCB of volume in use
: 190 1179 1
: 191 1180 1 OUTPUT PARAMETERS:
: 192 1181 1 NONE
: 193 1182 1
: 194 1183 1 IMPLICIT OUTPUTS:
: 195 1184 1 NONE
: 196 1185 1
: 197 1186 1 ROUTINE VALUE:
: 198 1187 1 NONE
: 199 1188 1
: 200 1189 1 SIDE EFFECTS:
: 201 1190 1 Volume dismounted if appropriate
: 202 1191 1
: 203 1192 1 --
: 204 1193 1
: 205 1194 2 BEGIN
: 206 1195 2
: 207 1196 2 LINKAGE DALLOC_DEV = JSB (REGISTER = 4, REGISTER = 5)
: 208 1197 2 : NOPRESERVE (3)
: 209 1198 2 : PRESERVF (2, 4, 5)
: 210 1199 2 : NOTUSED (6, 7, 8, 9, 10, 11);
: 211 1200 2
: 212 1201 2
: 213 1202 2 LOCAL J. | loop index
: 214 1203 2 RVT_LENGTH. | number of entries in RVT
: 215 1204 2 RVT : REF BBLOCK; | address of RVT (or UCB if not a set)
: 216 1205 2
: 217 1206 2
: 218 1207 2 EXTERNAL
: 219 1208 2 CTL$GL_PCB : ADDRESSING_MODE(GENERAL), ! PCB address
: 220 1209 2 CTL$GL_PHD : ADDRESSING_MODE(GENERAL), ! PHD address
: 221 1210 2 IOC$GL_AQBLIST : REF BBLOCK ADDRESSING_MODE(ABSOLUTE); ! AQB listhead
: 222 1211 2
: 223 1212 2 BIND_COMMON;
: 224 1213 2
: 225 1214 2 LINKAGE DEAP = JSB (REGISTER=0, REGISTER=1) : NOPRESERVE (2,3,4,5);
: 226 1215 2
: 227 1216 2
: 228 1217 2 EXTERNAL ROUTINE
```

```
; 229      1218 2    CONV_ACLOCK   : L_NORM,          convert/dequeue access lock.
; 230      1219 2    LOCK_COUNT    : L_NORM,          Determine count of locks granted.
; 231      1220 2    WAIT_FOR_AST : L_NORM NOVALUE ADDRESSING_MODE (GENERAL),
; 232      1221 2    ; exit thread until completion ast
; 233      1222 2    CONTINUE_THREAD: L_NORM NOVALUE ADDRESSING_MODE (GENERAL),
; 234      1223 2    ; completion ast to resume thread
; 235      1224 2    LOCK_IODB     : L_NORM,          lock I/O data base mutex
; 236      1225 2    UNLOCK_IODB   : L_NORM,          unlock I/O data base mutex
; 237      1226 2    DEQ_LOCK     : L_NORM,          dequeue a lock
; 238      1227 2    DEALLOCATE   : L_NORM,          deallocate dynamic memory
; 239      1228 2    SWITCH_CHANNEL: L_NORM,          switch channels to specified UCB
; 240      1229 2    SEND_ERRLOG  : L_NORM,          send message to error logger
; 241      1230 2    EXESDEAPGDSIZ: DEAP ADDRESSING_MODE (GENERAL),
; 242      1231 2    ; Deallocate paged pool.
; 243      1232 2    IOC$DALLOC_DMT: DALLOC_DEV ADDRESSING_MODE (GENERAL),
; 244      1233 2    ; deallocate device
; 245      1234 2    ACL_DELETEACL:           ; Delete & deallocate ACL segments
; 246      1235 2
; 247      1236 2
; 248      1237 2    ! Get the RVT address and iterate on the whole volume set, since deaccessing
; 249      1238 2    a multi-volume file could make several volumes eligible for dismount. If
; 250      1239 2    this is not a volume set we special case and exit.
; 251      1240 2
; 252      1241 2
; 253      1242 2    J = 1;
; 254      1243 2    RVT = .CURRENT_VCB[VCBSL_RVT];
; 255      1244 2    IF .RVT NEQ .CURRENT_UCB
; 256      1245 2    THEN RVT_LENGTH = .RVT[RVT$B_NVOLS];
; 257      1246 2
; 258      1247 2    DO
; 259      1248 3    BEGIN
; 260      1249 3
; 261      1250 3    ! Declare most locals here for substantial improvement in storage allocation.
; 262      1251 3
; 263      1252 3    LOCAL
; 264      1253 3    LOCKCOUNT   : INITIAL (0),    ! count of volume locks
; 265      1254 3    STS,          ! general status value
; 266      1255 3    LKS$S        : VECTOR [6],    ! lock status block
; 267      1256 3    AQB,          ! REF BBLOCK,       ! address of XQP AQB
; 268      1257 3    CACHE         : REF BBLOCK,       ! address of volume cache
; 269      1258 3    UCB,          ! REF BBLOCK,       ! local address of UCB
; 270      1259 3    ORB,          ! REF BBLOCK,       ! local address of ORB
; 271      1260 3    VCB,          ! REF BBLOCK,       ! local address of VCB
; 272      1261 3    FCB,          ! REF BBLOCK,       ! local address of FCB
; 273      1262 3    WCB,          ! REF BBLOCK;       ! local address of WCB
; 274      1263 3
; 275      1264 3    UCB = .RVT;          ! assume not volume set
; 276      1265 3
; 277      1266 3    IF .UCB NEQ .CURRENT_UCB ! get UCB if volume set
; 278      1267 3    THEN UCB = .VECTOR [RVT[RVT$L_UCBLST], .J-1];
; 279      1268 3
; 280      1269 3    ! First check the mark for dismount bit.
; 281      1270 3
; 282      1271 3
; 283      1272 3
; 284      1273 3    IF .UCB NEQ 0
; 285      1274 3    THEN IF .BBLOCK [UCB[UCBSL_DEVCHAR], !EV$V_DMT]
```

```
; 286 1275 3 THEN
; 287 1276 4 BEGIN
; 288 1277 4
; 289 1278 4 ! Do volume switch if necessary.
; 290 1279 4 !
; 291 1280 4
; 292 1281 4 IF .UCB NEQ .CURRENT UCB
; 293 1282 4 THEN SWITCH_CHANNEL T.UCB;
; 294 1283 4
; 295 1284 4 ! The volume is marked for dismount. The remainder of the tests and the
; 296 1285 4 ! dismount bit twiddling must be done interlocked.
; 297 1286 4
; 298 1287 4
; 299 1288 4 LOCK_IODB ();
; 300 1289 4 SET_IPL (IPL$_SYNCH);
; 301 1290 4
; 302 1291 4 ORB = .UCB[UCBSL_ORB];
; 303 1292 4 VCB = .UCB[UCBSL_VCB];
; 304 1293 4 IF .VCB[VCBSW_TRANS] NEQ 1
; 305 1294 4 THEN
; 306 1295 4 UNLOCK_IODB ()
; 307 1296 4 ELSE
; 308 1297 5 BEGIN
; 309 1298 5
; 310 1299 5 ! The volume is marked for dismount and idle. Set the dismount in progress
; 311 1300 5 bit to stop all further activity.
; 312 1301 5
; 313 1302 5
; 314 1303 5 UCB[UCBSV_DISMOUNT] = 1;
; 315 1304 5 UNLOCK_IODB ();
; 316 1305 5
; 317 1306 5 ! Make an error log entry to record the dismount.
; 318 1307 5 !
; 319 1308 5
; 320 1309 5 SEND_ERRLOG (0, .UCB);
; 321 1310 5
; 322 1311 5 ! Release the device as specified in the applicable dismount request
; 323 1312 5 ! by issuing either an IOS_UNLOAD or an IOS_AVAILABLE function.
; 324 1313 5 !
; 325 1314 5
; 326 1315 6 BEGIN
; 327 1316 6 LOCAL
; 328 1317 6 QIOSTAT,
; 329 1318 6 PTR : REF BBLOCK,
; 330 1319 6 SAVE_PRIV : VECTOR [4];
; 331 1320 6
; 332 1321 6 ! Save and restore PHY_IO privilege around the QIO.
; 333 1322 6 !
; 334 1323 6
; 335 1324 6 PTR = .CTL_SGL PCB;
; 336 1325 6 PTR [PCBSW_DIOCNT] = .PTR [PCBSW_DIOCNT] + 1;
; 337 1326 6 PTR [PCBSW_ASTCNT] = .PTR [PCBSW_ASTCNT] + 1;
; 338 1327 6 SAVE_PRIV [0] = .(PTR [PCBSQ_PRIV]);
; 339 1328 6 SAVE_PRIV [1] = .(PTR [PCBSQ_PRIV]+4);
; 340 1329 6
; 341 1330 6 BBLOCK [PTR [PCBSQ_PRIV], PRV$V_PHY_IO] = 1;
; 342 1331 6 BBLOCK [PTR [PCBSQ_PRIV], PRV$V_BYPASS] = 1;
```

```
: 343      1332 6
: 344      1333 6
: 345      1334 6
: 346      1335 6
: 347      1336 6
: 348      1337 6
: 349      1338 6
: 350      1339 6 ! Issue an unload function if unload was requested.
: 351      1340 6
: 352      1341 6
: P 1342 6
: P 1343 6
: P 1344 6
: P 1345 6
: P 1346 6
: P 1347 6
: P 1348 6
: P 1349 6
: 1350 6
: 1351 6
: P 1352 6
: P 1353 6
: P 1354 6
: P 1355 6
: P 1356 6
: P 1357 6
: P 1358 6
: P 1359 6
: 1360 6
: 1361 6
: 1362 5
: 1363 5
: 1364 5 ! If this is a shared mount, raise the device lock to PW to get the
: 1365 5 value block, and prepare for writing it back. If the device is not
: 1366 5 shared, the lock is already at EX. If the device is not cluster
: 1367 5 accessible, there is no lock.
: 1368 5
: 1369 5
: 1370 5
: 1371 5
: 1372 5
: THEN
: BEGIN
: P 1373 6
: STS = SENO (LKMODE = LCK$K_PMMODE,
:             LKSB  = LKSTS,
:             EFN   = EFN,
:             ASTADR = CONTINUE_THREAD,
:             ASTPRM = .BASE,
:             FLAGS  = LCK$M_CONVERT + LCK$M_SYNCSTS
:             + LCK$M_NOQUOTA);
: 1374 6
: 1375 6
: 1376 6
: 1377 6
: 1378 6
: 1379 6
: 1380 6
: 1381 6
: 1382 6
: 1383 6
: 1384 6
: 1385 6
: 1386 6
: 1387 6
: 1388 6 ! Determine whether this is the last (only) lock for this volume.
```

```
400 1389 6 !
401 1390 6
402 1391 6      LOCKCOUNT = LOCK_COUNT (.VCB [VCB$L_VOLLKID]);
403 1392 6      END
404 1393 5      ELSE
405 1394 5      LOCKCOUNT = 1;           ! always 1 if allocated.
406 1395 5
407 1396 5      Now relock the I/O database and finish the dismount.
408 1397 5      Mark the volume dismounted and disconnect the VCB from the UCB.
409 1398 5      The high bit of the dirseq is masked off. This tells RMS the lock
410 1399 5      is disarmed.
411 1400 5
412 1401 5
413 1402 5      LOCK_IODB ();
414 1403 5      (UCB[UCB$W_DIRSEQ])<15,1> = 0;
415 1404 5      BBLOCK [UCB[UCB$L_DEVCHAR], DEV$V_MNT] = 0;
416 1405 5      BBLOCK [UCB[UCB$L_DEVCHAR], DEV$V_DMT] = 0;
417 1406 5      BBLOCK [UCB[UCB$L_DEVCHAR], DEV$V_SWL] = 0;
418 1407 5      UCB[UCB$W_REF_C] = UCB[UCB$W_REF_C] - 1;
419 1408 5      UCB[UCB$V_DISMOUNT] = 0;
420 1409 5      UCB[UCB$L_VCB] = 0;
421 1410 5      ORB[ORB$L_SYS_PROT] = 0;
422 1411 5      ORB[ORB$L_OWN_PROT] = 0;
423 1412 5      ORB[ORB$L_GRP_PROT] = 0;
424 1413 5      ORB[ORB$L_WOR_PROT] = 0;
425 1414 5      ORB[ORB$L_OWNER] = 0;
426 1415 5
427 1416 5      Decrement the mount count on the AQB. If it goes to zero, remove
428 1417 5      this AQB from the list and remember to deallocate it after we're done
429 1418 5      flushing buffers a little further on.
430 1419 5
431 1420 5
432 1421 5      AQB = .VCB [VCB$L_AQB];
433 1422 5      IF (AQB[AQBSB_MNTCNT] = .AQB[AQBSB_MNTCNT] - 1) NEQ 0
434 1423 5      THEN
435 1424 5      AQB = 0
436 1425 5      ELSE
437 1426 6      BEGIN
438 1427 6      LOCAL P : REF BBLOCK;
439 1428 6
440 1429 6      P = .IOC$GL_AQBLIST;
441 1430 6      IF .P EQL .AQB
442 1431 6      THEN
443 1432 6      IOC$GL_AQBLIST = .AQB[AQBSL_LINK]
444 1433 6      ELSE
445 1434 7      BEGIN
446 1435 7      UNTIL .P[AQBSL_LINK] EQL .AQB
447 1436 7      DO P = .P[AQBSL_LINK];
448 1437 7      P[AQBSL_LINK] = .AQB[AQBSL_LINK];
449 1438 6      END;
450 1439 5      END;
451 1440 5
452 1441 5      Deallocate the remaining file control blocks and caches.
453 1442 5
454 1443 5
455 1444 5      UNTIL REMQUE (.VCB[VCB$L_FCBFL], FCB)
456 1445 5      DO
```

```
: 457      1446 6          BEGIN
: 458      1447 6          FCB[FCBSW_REFCNT] = 0;           ! force deq on conv_acclock
: 459      1448 6          CONV_ACLOCK(0, .FCB);           ! deq access lock, if any
: 460      1449 6          IF .BBLOCK[FCB[FCBSR_ORB], ORBSV_ACL_QUEUE]
: 461      1450 6          THEN ACL_DELETEACL(FCB[FCBSL_AC[FL]], 0); ! Delete the ACL
: 462      1451 6          UNTIL REMOVE(.FCB[FCBSL_WLFL], WCB) ! deallocate all window
: 463      1452 6          DO DEALLOCATE(.WCB);           ! segments
: 464      1453 6          DEALLOCATE(.FCB);           ! release all FCB's
: 465      1454 5          END;
: 466      1455 5
: 467      1456 5          CACHE = .VCB[VCBSL_CACHE];
: 468      1457 5          IF .BBLOCK[.CACHE[VCASL_FIDCACHE], VCASL_FIDCLKID] NEQ 0
: 469      1458 5          THEN DEQ_LOCK(.BBLOCK[.CACHE[VCASL_FIDCACHE], VCASL_FIDCLKID]);
: 470      1459 5          IF .BBLOCK[.CACHE[VCASL_EXTCACHE], VCASL_EXTCLKID] NEQ 0
: 471      1460 5          THEN DEQ_LOCK(.BBLOCK[.CACHE[VCASL_EXTCACHE], VCASL_EXTCLKID]);
: 472      1461 5          DEALLOCATE(.VCB[VCBSL_CACHE]); ! release the cache block
: 473      1462 5
: 474      1463 5          CACHE = .VCB[VCBSL_QUOCACHE];
: 475      1464 5          IF .CACHE NEQ 0           ! release quota cache if present
: 476      1465 5          THEN
: 477      1466 6          BEGIN
: 478      1467 6          IF .CACHE[VCASL_QUOCLKID] NEQ 0
: 479      1468 6          THEN DEQ_LOCK(.CACHE[VCASL_QUOCLKID]);
: 480      1469 6          DEALLOCATE(.VCB[VCBSL_QUOCACHE]);
: 481      1470 5          END;
: 482      1471 5
: 483      1472 5          ! Dequeue the volume lock.
: 484      1473 5
: 485      1474 5
: 486      1475 5          DEQ_LOCK(.VCB[VCBSL_VOLLKID]);
: 487      1476 5
: 488      1477 5          IF .RVT NEQ .CURRENT_UCB
: 489      1478 5          THEN
: 490      1479 6          BEGIN
: 491      1480 6          VECTOR[RVT[RVT$L_UCLST], .VCB[VCBSW_RVN]-1] = 0;
: 492      1481 6          RVT[RVT$W_REFCT] = .RVT[RVT$W_REFCT] - 1;
: 493      1482 6          IF .RVT[RVT$W_REFCT] EQ 0
: 494      1483 6          THEN
: 495      1484 7          BEGIN
: 496      1485 7
: 497      1486 7          DEQ_LOCK(.RVT[RVT$L_STRUCLKID]);
: 498      1487 7
: 499      1488 7          ! Dequeue blocking lock and disable blocking check on exit.
: 500      1489 7
: 501      1490 7
: 502      1491 7          IF .RVT[RVT$L_BLOCKID] NEQ 0
: 503      1492 7          THEN DEQ_LOCK(.RVT[RVT$L_BLOCKID]);
: 504      1493 7          BLOCK_CHECK = 0;
: 505      1494 7
: 506      1495 7          DEALLOCATE(.RVT);
: 507      1496 6          END;
: 508      1497 6          END
: 509      1498 5          ELSE
: 510      1499 6          BEGIN
: 511      1500 6          IF .VCB[VCBSL_BLOCKID] NEQ 0
: 512      1501 6          THEN DEQ_LOCK(.VCB[VCBSL_BLOCKID]);
: 513      1502 6          BLOCK_CHECK = 0;
```

```
: 514      1503 5      END;  
: 515      1504 5  
: 516      1505 5      DEALLOCATE (.VCB);           ! release the VCB  
: 517      1506 5  
: 518      1507 5      | If the device lock exists, now demote it as appropriate (to CR if  
: 519      1508 5      | the device is not allocated, to EX otherwise). Clear the value  
: 520      1509 5      | block if this is the final dismount.  
: 521      1510 5  
: 522      1511 5  
: 523      1512 5      IF .LKSTS [1] NEQ 0  
: 524      1513 5      THEN  
: 525      1514 6      BEGIN  
: 526      1515 6      LOCAL LKFLGS;  
: 527      1516 6      LKFLGS = LCK$M_CONVERT + LCK$M_CVTSYS  
: 528      1517 6      + LCK$M_SYNCSTS + LCK$M_NCQUOTA;  
: 529      1518 6  
: 530      1519 6  
: 531      1520 6      IF .LOCKCOUNT EQL 1  
: 532      1521 7      THEN  
: 533      1522 7      BEGIN  
: 534      1523 7      LKFLGS = .LKFLGS + LCK$M_VALBLK;  
: 535      1524 7      LKSTS [2] = 0;  
: 536      1525 7      LKSTS [3] = 0;  
: 537      1526 7      LKSTS [4] = 0;  
: 538      1527 6      LKSTS [5] = 0;  
: 539      1528 6      END;  
P 540      1529 6      STS = $ENQ (LKMODE = IF .UCB [UCBSL PID] NEQ 0  
P 541      1530 6      THEN LCK$K_EXMODE  
P 542      1531 6      ELSE LCK$K_CRMODE,  
P 543      1532 6      LKSB = LKSTS,  
P 544      1533 6      EFN = EFN,  
P 545      1534 6      FLAGS = .LKFLGS);  
P 546      1535 6      IF NOT .STS  
P 547      1536 6      OR NOT .LKSTS  
P 548      1537 6      THEN BUG_CHECK (XQPERR, FATAL, 'Unexpected lock manager error');  
P 549      1538 5      END;  
P 550      1539 5      | Call IOC$DALLOC_DMT routine to deallocate the device when appropriate.  
P 551      1540 5  
P 552      1541 5  
P 553      1542 5  
P 554      1543 5      IOC$DALLOC_DMT (.CTL$GL_PCB, .UCB);  
P 555      1544 5  
P 556      1545 5      UNLOCK_IODB ();  
P 557      1546 5  
P 558      1547 5      IF .AOB NEQ 0  
P 559      1548 5      THEN  
P 560      1549 6      BEGIN  
P 561      1550 6      LOCAL P : REF BBLOCK;  
P 562      1551 6      P = .AOB [AOBSL_BUFCACHE];  
P 563      1552 6      EXEC$DEAPGDSIZ (.P, .P [F1IBCSL_REALSIZE]);  
P 564      1553 6      DEALLOCATE (.AOB);  
P 565      1554 5      END;  
P 566      1555 5  
P 567      1556 4      END;          ! end of dismount processing  
P 568      1557 4  
P 569      1558 3      END;          ! end of dismount condition  
P 570      1559 3
```

```

: 571 1560 3 IF .RVT EQL .CURRENT_UCB THEN EXITLOOP;
: 572 1561 3
: 573 1562 3 J = .J + 1;                                ! bump loop index
: 574 1563 3 END                                     ! end of multi-volume loop
: 575 1564 2 UNTIL .J GTRU .RVT_LENGTH;
: 576 1565 2
: 577 1566 1 END;                                    ! end of routine CHECK_DISMOUNT

```

```

.TITLE CHKDMO
.IDENT \V04-000\

.EXTRN CTL$GL_PCB, CTL$GL_PHD
.EXTRN IOCSGL_AQBLIST, CONV_ACLOCK
.EXTRN LOCK_COUNT, WAIT_FOR_AST
.EXTRN CONTINUE_THREAD
.EXTRN LOCK_IODB, UNLOCK_IODB
.EXTRN DEQ_LOCK, DEALLOCATE
.EXTRN SWITCH_CHANNEL, SEND_ERRLOG
.EXTRN EXESDEAPGDSIZ, IOCSDA$ALLOC_DMT
.EXTRN ACL_DELETEACL, SYSSQ10
.EXTRN SYSSENQ, BUGS_XOPERR

.PSECT SLOCKEDC1$, NOWRT, 2

        OBFC 00000
.ENTRY CHECK_DISMOUNT, Save R2,R3,R4,R5,R6,R7,R8,- : 1161
R9,R1T
5E      94 30 C2 00002
59      94 AA 9E 00005
58      01 D0 00009
50      98 AA D0 0000C
56      20 A0 D0 00010
69      56 D1 00014
6E      04 13 00017
       08 A6 9A 00019
       04 AE D4 0001D 1$: MOVZBL 11(RVT), RVT_LENGTH : 1245
55      56 D0 00020
69      55 D1 00023
       05 13 00026
       55 D0 00028
       55 D5 0002D 2$: CLRL 1248
27      3A A5 05 E1 00031
       69 55 D1 00036
       07 13 00039
       55 DD 0003B
       01 FB 0003D 3$: CALLS #1, SWITCH CHANNEL : 1288
0000G  CF 00 FB 00042 3$: CALLS #0, LOCK_IODB : 1289
0000G  CF 08 DA 00047
12      MTPR #8, #18
54      1C A5 D0 0004A
53      34 A5 D0 0004E
01      OC A3 B1 00052
       08 13 00056
       00 FB 00058
0000G  CF 0281 31 0005D 4$: CALLS #0, UNLOCK_IODB : 1295
       66 A5 10 88 00060 5$: BRW 34$ : 1303
0000G  CF 00 FB 00064 5$: BISB2 #16, 102(UVB) : 1304

```


04	AE		01	D0	00138	12\$:	MOVL	#1, LOCKCOUNT	: 1394	
0000G	CF		00	FB	0013C	13\$:	CALLS	#0, LOCK_IODB	: 1402	
00AD	C5	80	8F	8A	00141		BICB2	#128, 173(UCB)	: 1403	
3A	A5	0228	8F	AA	00147		BICW2	#552, 58(UCB)	: 1406	
		5C	A5	B7	0014D		DECW	92(UCB)	: 1407	
66	A5		10	8A	00150		BICB2	#16, 102(UCB)	: 1408	
		34	A5	D4	00154		CLRL	52(UCB)	: 1409	
		18	A4	7C	00157		CLRQ	24(ORB)	: 1410	
		20	A4	7C	0015A		CLRQ	32(ORB)	: 1412	
		64	D4	D0	0015D		CLRL	(ORB)	: 1414	
57		10	A3	D0	0015F		MOVL	16(VCB), AQB	: 1421	
50		08	A7	9A	00163		MOVZBL	11(AQB), R0	: 1422	
			50	D7	00167		DECL	R0		
08	A7		50	90	00169		MOVBL	R0, 11(AQB)		
			50	D5	0016D		TSTL	R0		
			04	13	0016F		BEQL	14\$		
			57	D4	00171		CLRL	AQB	: 1424	
			27	11	00173		BRB	17\$		
		50 00000000G	9F	D0	00175	14\$:	MOVL	#IOC\$GL_AQBLIST, P	: 1429	
		57	50	D1	0017C		CMPL	P, AQB	: 1430	
			0A	12	0017F		BNEQ	15\$		
00000000G	9F	10	A7	D0	00181		MOVL	16(AQB), #IOC\$GL_AQBLIST	: 1432	
			11	11	00189		BRB	17\$		
		57	10	A0	0018B	15\$:	CMPL	16(P), AQB	: 1435	
			06	13	0018F		BEQL	16\$		
50		10	A0	D0	00191		MOVL	16(P), P	: 1436	
			F4	11	00195		BRB	15\$		
10	A0	10	A7	D0	00197	16\$:	MOVL	16(AQB), 16(P)	: 1437	
52		00	B3	OF	0019C	17\$:	REMQUE	#0(VCB), FCB	: 1444	
			34	1D	001A0		BVS	20\$		
			18	A2	B4	001A2	CLRW	24(FCB)	: 1447	
			52	DD	001A5		PUSHL	FCB	: 1448	
			7E	D4	001A7		CLRL	-(SP)		
08	0000G	CF	02	FB	001A9		CALLS	#2, CONV_ACLOCK		
	63	A2	01	E1	001AE		BBC	#1, 99(FCB), 18\$: 1449	
			7E	D4	001B3		CLRL	-(SP)	: 1450	
	0000G	CF	0080	C2	9F	001B5	PUSHAB	128(FCB)		
	54		02	FB	001B9		CALLS	#2, ACL_DELETEACL		
		10	B2	OF	001BE	18\$:	REMQUE	#16(FCB), WCB	: 1451	
			09	1D	001C2		BVS	19\$		
			54	DD	001C4		PUSHL	WCB	: 1452	
0000G	CF		01	FB	001C6		CALLS	#1, DEALLOCATE		
			F1	11	001CB		BRB	18\$		
	0000G	CF		52	DD	001CD	19\$:	PUSHL	FCB	: 1453
			01	FB	001CF		CALLS	#1, DEALLOCATE		
			C6	11	001D4		BRB	17\$		
52		58	A3	D0	001D6	20\$:	MOVL	88(VCB), CACHE	: 1444	
50			62	DD	001DA		MOVL	(CACHE), R0	: 1456	
		04	A0	D5	001DD		TSTL	4(R0)	: 1457	
			08	A3	001E0		BEQL	21\$		
	0000G	CF	04	A0	DD	001E2	PUSHL	4(R0)	: 1458	
	50		01	FB	001E5		CALLS	#1, DEQ_LOCK	: 1459	
			04	A2	D0	001EA	21\$:	MOVL	4(CACHE), R0	
			0C	A0	D5	001EE	TSTL	12(R0)		
			08	13	001F1		BEQL	22\$		
0000G	CF	0C	A0	DD	001F3		PUSHL	12(R0)	: 1460	
			01	FB	001F6		CALLS	#1, DEQ_LOCK		

0000G	CF	58	A3	DD	001FB	22\$:	PUSHL	88(VCB)	1461	
		5C	A3	DD	001FE		CALLS	#1, DEALLOCATE	:	
		5C	A3	DD	00203		MOVL	92(VCB), CACHE	1463	
		04	A2	D5	00207		BEQL	24\$	1464	
		04	A2	D5	00209		TSTL	4(CACHE)	1467	
		04	A2	DD	0020C		BEQL	23\$:	
		04	A2	DD	0020E		PUSHL	4(CACHE)	1468	
0000G	CF	5C	A3	DD	00211	23\$:	CALLS	#1, DEQ_LOCK	:	
		5C	A3	DD	00216		PUSHL	92(VCB)	1469	
0000G	CF	7C	A3	DD	0021E	24\$:	CALLS	#1, DEALLOCATE	:	
		01	FB	00221		PUSHL	124(VCB)	1475		
0000G	CF	69	56	D1	00226		CALLS	#1, DEQ_LOCK	:	
		2D	13	00229		CMPL	RVT, (R9)	1477		
		0E	A3	3C	0022B		BEQL	26\$:	
		40	A640	D4	0022F		MOVZWL	14(VCB), R0	1480	
		04	A6	B7	00233		CLRL	64(RVT)[R0]	:	
			31	12	00236		DECW	4(RVT)	1481	
			66	DD	00238		BNEQ	28\$	1482	
0000G	CF	24	A6	D5	0023A		PUSHL	(RVT)	1486	
		08	A6	D5	0023F		CALLS	#1, DEQ_LOCK	:	
		24	A6	DD	00242		TSTL	36(RVT)	1491	
0000G	CF	24	A6	DD	00244		PUSHL	36(RVT)	:	
		01	FB	00247		CALLS	#1, DEQ_LOCK	1492		
		A7	AA	94	0024C	25\$:	CLRB	-89(BASE)	:	
			56	DD	0024F		PUSHL	RVT	1493	
0000G	CF	01	FB	00251		CALLS	#1, DEALLOCATE	1495		
		11	11	00256		BRB	28\$:		
		50	008C	C3	DD	00258	26\$:	MOVL	140(VCB), R0	1477
		07	13	0025D		BEQL	27\$:		
		50	DD	0025F		PUSHL	R0	1500		
0000G	CF	01	FB	00261		CALLS	#1, DEQ_LOCK	:		
		A7	AA	94	00266	27\$:	CLRB	-89(BASE)	1502	
0000G	CF	53	DD	00269	28\$:	PUSHL	VCB	1505		
		01	FB	0026B		CALLS	#1, DEALLOCATE	:		
		1C	AE	D5	00270		TSTL	LKSTS+4	1512	
		41	13	00273		BEQL	33\$:		
01		6A	8F	9A	00275		MOVZBL	#106, LKFLGS	1517	
		04	AE	D1	00279		CMPL	LOCKCOUNT, #1	1519	
		08	12	0027D		BNEQ	29\$:		
		50	D6	0027F		INCL	LKFLGS	1522		
		20	AE	7C	00281		CLRQ	LKSTS+8	1523	
		28	AE	7C	00284	29\$:	CLRQ	LKSTS+16	1525	
		7E	7C	00287		CLRQ	-(SP)	1534		
		7E	7C	00289		CLRQ	-(SP)	:		
		7E	7C	0028B		CLRQ	-(SP)	:		
		7E	D4	0028D		CLRL	-(SP)	:		
		38	50	DD	0028F		PUSHL	LKFLGS	:	
		2C	AE	9F	00291		PUSHAB	LKSTS	:	
		04	A5	D5	00294		TSTL	44(UCB)	:	
		05	DD	00297		BEQL	30\$:		
		02	11	00299		PUSHL	#5	:		
		01	DD	0029D	30\$:	BRB	31\$:		
00000000G	00	08	FB	002A1	31\$:	PUSHL	#1	:		
		58	50	DD	002A8		CALLS	#11, SYS\$ENQ	:	
						MOVL	R0, STS	:		

04		5B E9 002AB	BLBC	STS 32\$; 1535
04	18	AE E8 002AE	BLBS	LKSFS, 33\$; 1536
		FEFF 002B2	32\$:	BUGW	; 1537
		0000* 002B4		.WORD <BUGS_XOPERR!4>	
54	00000000G	00 D0 002B6	33\$:	MOVL CTL\$GE[PCB, R4	; 1543
00000000G	00 16 002BD			JSB IOCSDA[LOC_DMT	
0000G CF	00 FB 0^2C3		CALLS #0, UNLOCK_IODB	; 1545	
	57 D5 002C8		TSTL AQB	; 1547	
	15 13 002CA		BEQL 34\$		
50	18 A7 D0 002CC		MOVL 24(AQB), P	; 1551	
51	OC A0 D0 002D0		MOVL 12(P), R1	; 1552	
00000000G	00 16 002D4		JSB EXESDÉAPGDSIZ		
0000G CF	57 DD 002DA		PUSHL AQB	; 1553	
69	01 FB 002DC		CALLS #1, DEALLOCATE		
	56 D1 002E1	34\$:	CMPL RV\$, (R9)	; 1560	
	0A 13 002E4		BEQL 35\$		
	58 D6 002E6		INCL J	; 1562	
6E	58 D1 002E8		CMPL J, RVT_LENGTH	; 1564	
	03 1A 002EB		BGTRU 35\$		
	FD2D 31 002ED		BRW 1\$		
	04 002F0 35\$:		RET	; 1566	

: Routine Size: 753 bytes, Routine Base: \$LOCKEDC1\$ + 0000

```
579 1567 1 GLOBAL ROUTINE UPDATE_DIRSEQ : L_NORM =
580 1568 1
581 1569 1 !++
582 1570 1
583 1571 1 FUNCTIONAL DESCRIPTION:
584 1572 1
585 1573 1 This routine bumps the directory sequence count in the UCB to invalidate
586 1574 1 RMS directory caches on the volume.
587 1575 1
588 1576 1 CALLING SEQUENCE:
589 1577 1 UPDATE_DIRSEQ ()
590 1578 1
591 1579 1 INP T PARAMETERS:
592 1580 1 NONE
593 1581 1
594 1582 1 IMPLICIT INPUTS:
595 1583 1 CURRENT_UCB: UCB of device in use
596 1584 1 CURRENT_RVT:
597 1585 1 NVOLS: number of volumes in volume set
598 1586 1 UCBLST: addresses of UCB's in volume set
599 1587 1
600 1588 1
601 1589 1 OUTPUT PARAMETERS:
602 1590 1 NONE
603 1591 1
604 1592 1 IMPLICIT OUTPUTS:
605 1593 1 directory sequence count incremented
606 1594 1
607 1595 1 ROUTINE VALUE:
608 1596 1 1
609 1597 1
610 1598 1 SIDE EFFECTS:
611 1599 1 NONE
612 1600 1
613 1601 1 --
614 1602 1
615 1603 2 BEGIN
616 1604 2
617 1605 2 BIND_COMMON:
618 1606 2
619 1607 2 EXTERNAL ROUTINE
620 1608 2 ALLOCATION_LOCK : L_NORM NOVALUE,
621 1609 2 ALLOCATION_UNLOCK : L_NORM NOVALUE,
622 1610 2 SWITCH_VOLUME : L_NORM NOVALUE,
623 1611 2 QEX_N_CANCEL : L_NORM;
624 1612 2
625 1613 2 LOCAL
626 1614 2 CURRVN,
627 1615 2 HAD_LOCK,
628 1616 2 VCB : REF BBLOCK, ! VCB address
629 1617 2 UCB : REF BBLOCK; ! UCB address
630 1618 2
631 1619 2 ! Iterate over the mounted volumes of a volume set if there is one.
632 1620 2
633 1621 2
634 1622 2 CURRVN = .CURRENT_RVN;
635 1623 2
```

```

: 636 1624 2 HAD_LOCK = 0;
: 637 1625 2
: 638 1626 2 IF .LB_LOCKID [0] NEQ 0
: 639 1627 2 THEN
: 640 1628 2 HAD_LOCK = 1
: 641 1629 2 ELSE
: 642 1630 2 ALLOCATION_LOCK ();
: 643 1631 2
: 644 1632 2 UCB = .CURRENT_UCB;
: 645 1633 2 IF .CURRENT_RVT NEQ .UCB
: 646 1634 2 THEN
: 647 1635 3 BEGIN
: 648 1636 3 INCR J FROM 1 TO .CURRENT_RVT[RVT$B_NVOLS]
: 649 1637 3 DO
: 650 1638 4 BEGIN
: 651 1639 4 VCB = 0;
: 652 1640 4 UCB = .VECTOR [CURRENT_RVT[RVT$L_UCBLST], .J-1];
: 653 1641 4 IF .UCB NEQ 0
: 654 1642 4 THEN
: 655 1643 4 IF (VCB = .UCB [UCB$L_VCB]) NEQ 0
: 656 1644 4 THEN
: 657 1645 5 BEGIN
: 658 1646 5 SWITCH_VOLUME (.J);
: 659 1647 5 QEX_N_CANCEL (.LB_LOCKID [0]);
: 660 1648 4 END;
: 661 1649 3
: 662 1650 3 END;
: 663 1651 3
: 664 1652 2 ELSE
: 665 1653 2 QEX_N_CANCEL (.LB_LOCKID [0]);
: 666 1654 2
: 667 1655 2 IF NOT .HAD_LOCK
: 668 1656 2 THEN
: 669 1657 2 ALLOCATION_UNLOCK ();
: 670 1658 2
: 671 1659 2 RETURN 1;
: 672 1660 2
: 673 1661 1 END;

```

! end of routine UPDATE_DIRSEQ

			.EXTRN ALLOCATION_LOCK	: 1567
			.EXTRN ALLOCATION_UNLOCK	: 1622
			.EXTRN SWITCH_VOLUME, QEX_N_CANCEL	: 1624
			.ENTRY UPDATE_DIRSEQ, Save R2,R3,R4,R5,R6,R7	: 1626
	57	A0 00FC 00000	MOV _L -96(BASE), CURRVN	: 1628
		AA D0 00002	CLRL HAD_LOCK	
	6C	56 D4 00006	TSTL 108(BASE)	
		AA D5 00008	BEQL 1\$	
	56	05 13 0000B	MOVL #1, HAD_LOCK	
		01 D0 0000D	BRB 2\$	
0000G	C9	00 FB 00012	CALLS #0, ALLOCATION_LOCK	: 1630
	53	AA D0 00017	MOV _L -108(BASE), UCB	: 1632
	50	94 AA D0 0001B	MOV _L -100(BASE), R0	: 1633
	53	50 D1 0001F	CMPL R0, UCB	
		37 13 00022	BEQL 5\$	

	55	08	A0	9A	00024	MOVZBL	11(R0), R5	: 1636
		52	D4	00028	CLRL	J	: :	
		22	11	0002A	BRB	4\$: :	
	50	9C	BA42	DE	0002C	3\$: CLRL	VCB	1639
	53	40	A0	DD	00033	MOVAL	8-100(BASE)[J], R0	1640
	54	34	A3	DD	00037	MOVL	64(R0), UCB	1641
			OF	13	00039	BEQL	4\$	1643
			52	DD	0003D	MOVL	52(UCB), VCB	1646
			01	FB	00041	PUSHL	J	1647
DA	0000G	CF	6C	AA	DD	00046	CALLS #1, SWITCH_VOLUME	1647
	0000G	CF		01	FB	00049	PUSHL 108(BASE)	1647
	52			55	F3	0004E	4\$: CALLS #1, QEX_N_CANCEL	1636
	0000G	CF		57	DD	00052	AOBLEQ R5, J, 3\$	1650
	0000G	CF		01	FB	00054	PUSHL CURRVN	1633
	0000G	CF		08	11	00059	CALLS #1, SWITCH_VOLUME	1653
	0000G	CF	6C	AA	DD	0005B	BRB 6\$	1655
	05			01	FB	0005E	PUSHL 108(BASE)	1657
	0000G	CF		56	E8	00063	CALLS #1, QEX_N_CANCEL	1659
	50			00	FB	00066	BLBS HAD_LOCK, 7\$	1661
			01	DD	0006B	CALLS #0, ALLOCATION_UNLOCK		
			04	0006E	7\$: MOVL #1, R0	RET		

: Routine Size: 111 bytes. Routine Base: \$LOCKEDC1\$ + 02F1

```
: 674 1662 1
: 675 1663 1 END
: 676 1664 0 ELUDOM
```

PSECT SUMMARY

Name	Bytes	Attributes
\$LOCKEDC1\$	864	NOVEC,NOWRT, RD, EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

Library Statistics

File	----- Symbols -----	Pages Mapped	Processing Time
	Total Loaded Percent		
\$_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619 84 0	1000	00:02.0

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:[CHKDMO/OBJ=OBJ\$:[CHKDMO MSRC\$:[CHKDMO/UPDATE=(ENH\$:[CHKDMO)

: Size: 864 code + 0 data bytes
: Run Time: 00:40.4
: Elapsed Time: 01:11.8
: Lines/CPU Min: 2473
: Lexemes/CPU-Min: 45046
: Memory Used: 405 pages
: Compilation Complete

0168 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

